Efficient Rule-Based Inferencing for OWL EL

Markus Krötzsch
Oxford University Computing Laboratory
markus.kroetzsch@comlab.ox.ac.uk

Abstract

We review recent results on inferencing for $\mathcal{SROEL}(\times)$, a description logic that subsumes the main features of the W3C recommendation OWL EL. Rule-based deduction systems are developed for various reasoning tasks and logical sublanguages. Certain feature combinations lead to increased space upper bounds for materialisation, suggesting that efficient implementations are easier to obtain for suitable fragments of OWL EL.

1 Introduction

Created in 2004 and updated in 2009, the Web Ontology Language OWL is a prominent knowledge representation standard of the World Wide Web Consortium [OWL, 2009]. Its *Direct Semantics* is based on description logics (DLs) that have a long tradition in knowledge representation and reasoning [Baader *et al.*, 2007]. To cater for a wide range of practical needs, OWL has been based on the particularly expressive DL $\mathcal{SROIQ}(D)$, but the resulting high complexity of reasoning is prohibitive in many applications. To solve this conflict, OWL introduced three lightweight *profiles* EL, QL, and RL as sublanguages of OWL [Motik *et al.*, 2009].

OWL RL is conceived as a "rule-based" OWL fragment that suggests inferencing with bottom-up materialisation rules. OWL QL was designed as a "query language" to support Ontology-Based Data Access, where inferencing is implemented by suitable query rewriting. OWL EL in turn is intended for conceptual modelling, a typical example being the medical ontology SNOMED CT that has no instance data but about 3×10^5 classes [James and Spackman, 2008]. While there are well-documented algorithms and various implementations for inferencing in RL and QL, a first comprehensive algorithm for EL has only been published recently [Krötzsch, 2010]. Moreover, this work also studies the relative "difficulty" of inferencing with some features in EL. Both aspects are summarised in this invited paper.

Semantically, OWL EL is an extension of the DL $\mathcal{EL}^{++}$ [Baader *et al.*, 2005]. This type of DLs is based on *conjunction* and *existential quantification*, allowing statements like $\mathsf{ColourBlind} \sqcap \mathsf{Female} \sqsubseteq \exists \mathsf{hasFather}.\mathsf{ColourBlind}$ ("Colour blind women must have a colour blind father"). $\mathcal{EL}^{++}$ also supports *role chains* (e.g. $\mathsf{hasParent} \circ \mathsf{hasBrother} \sqsubseteq \mathsf{hasUncle}$) and *nominal classes* (classes with a single named element, like italy in $\mathsf{Italian} \sqsubseteq \exists \mathsf{citizenOf}.\{\mathsf{italy}\}$). Moreover, one can add some forms of *range restrictions*, e.g. to state that Male is the range of $\mathsf{hasFather}$ [Baader *et al.*, 2008].

OWL EL further allows *local reflexivity* to model classes of individuals with some relation to themselves, e.g. to state $\mathsf{Narcist} \sqsubseteq \exists \mathsf{adores}.\mathsf{Self}$, and the *universal (top) role U* that relates all individuals. Both features can be combined with role chains to indirectly model *concept products* such as $\mathsf{Elephant} \times \mathsf{Mouse} \sqsubseteq \mathsf{biggerThan}$. We collect these features in the DL $\mathcal{SROEL}(\times)$ as defined in Section 2. One can also allow *role conjunctions* (a generalisation of OWL property disjointness), omitted here for reasons of space and since they are not part of OWL EL [Krötzsch, 2010]. OWL EL further includes support for *datatype properties* corresponding to concrete roles in DL. We omit this aspect here as datatypes in OWL EL can largely be treated like abstract classes.

Our contribution is twofold: (1) we introduce rule-based reasoning procedures for $\mathcal{SROEL}(\times)$, and (2) we show that some features of EL have a negative effect on the efficiency of such procedures. The significance of (1) is to provide the first comprehensive EL algorithm that copes with concept products and local reflexivity. We discuss cacluli both for instance retrieval (Section 3) and for classification (Section 4). Our use of rules to express deductions emphasises that implementations for EL can follow similar patterns as implementations for RL, even though the use of our rules in a general-purpose rule engine would hardly be as efficient as a specifically optimised algorithm.

All the calculi we present run in polynomial time, so it is not clear how to differentiate their "efficiency" (2). To this end, we consider the amounts of intermediate facts that correct rule-based bottom-up algorithms must compute (Section 5). Formally, we use the simple rule language Datalog, and we consider the required *minimal arity* of derived predicates in rule systems that are sound and complete for fragments of $\mathcal{SROEL}(\times)$. Since upper space bounds for Datalog are exponential in the *arity* of inferred predicates, our goal is to find materialisation calculi where these arities are low. A maximal arity of 2, e.g., establishes a quadratic upper bound on the number of derived facts. We find that this is enough for classification in $\mathcal{SROEL}(\times)$ fragments without role chains and nominals. But adding either feature to the DL increases the required arity by one, so $\mathcal{SROEL}(\times)$ does not admit any sound and complete classification calculus of arity below 4.

We obtain these results by studying possible derivations in Datalog. Our methods are novel but have some relations to studies on Datalog *width* and *parameterised complexity* that we outline in Section 6. Our optimality results refer to the syntactic form of rule-based algorithms, and have no immediate complexity-theoretic implications. Yet, experience shows that the problem of space efficiency w.r.t. nominals and role chains is relevant in practice, and it is not surprising that existing implementations tend to omit one or more of these features [Baader *et al.*, 2006; Delaitre and Kazakov, 2009]. Our results immediately apply to these works since it is not hard to state their inference rules in Datalog. In particular, our results imply that the algorithm of Baader *et al.* [2005] is incomplete in the presence of nominals.

Technical details and proofs are found in [Krötzsch, 2010].

2 The DL $\mathcal{SROEL}(\times)$ and Datalog

We chiefly summarise the required basic notions from DL and Datalog; see [Krötzsch, 2010] for formal definitions, and [Hitzler *et al.*, 2009] for a textbook introduction on the relation of DL and OWL. The DL $\mathcal{SROEL}(\times)$ is based on three disjoint finite sets of *individual names* $\mathbf{N_I}$, *concept names* $\mathbf{N_C}$, and *role names* $\mathbf{N_R}$. Semantically, these parts are interpreted as individuals, sets of individuals, and binary relations. The set $\mathbf{C}$ of $\mathcal{SROEL}(\times)$ *concepts* is given as

$$\mathbf{C} ::= \top \mid \bot \mid \mathbf{N_C} \mid \mathbf{C} \sqcap \mathbf{C} \mid \exists \mathbf{N_R}.\mathbf{C} \mid \exists \mathbf{N_R}.\mathsf{Self} \mid \{\mathbf{N_I}\}.$$

Concepts are interpreted as sets: $\top/\bot$ as the whole/empty set, conjunctions $\sqcap$ as a set intersection, and existential restrictions as sets of individuals with some particular role successor. Nominals $\{a\}$ encode singleton sets. Now a $\mathcal{SROEL}(\times)$ *axiom* can be an *assertion* $C(a)$ or $R(a, b)$, a *general concept inclusion* $C \sqsubseteq D$, or a *role inclusion* of one of the forms $R \sqsubseteq T$, $R \circ S \sqsubseteq T$, $C \times D \sqsubseteq T$, $R \sqsubseteq C \times D$ where $C, D \in \mathbf{C}$, $R, S, T \in \mathbf{N_R}$, $a, b \in \mathbf{N_I}$. Semantically, each axiom describes an according set-theoretic relationship, where $\circ$ encodes relational composition and $\times$ stands for the Cartesian product. Concept products allow us to define the universal (top) role U with an axiom $\top \times \top \sqsubseteq U$, while the empty (bottom) role N can be defined using $\exists N.\top \sqsubseteq \bot$. Knowledge bases KB are sets of $\mathcal{SROEL}(\times)$ axioms that satisfy some additional properties regarding *simplicity or roles* and *admissibility of range restrictions*. A restriction to *regular* role inclusions as in OWL DL (and OWL EL) is not needed. Entailment of $\mathcal{SROEL}(\times)$ is defined model theoretically, as usual.

Our formalisation of inferencing calculi is based on *Datalog* which we consider as standard first-order Horn logic without function symbols. As in deductive databases, we distinguish extensional (EDB) and intensional (IDB) predicates. EDB predicates can only occur in ground (variable-free) facts, and in premises of rules (bodies), so they are merely used to encode the original input. IDB predicates can occur without restrictions and are used for all derivations.

3 Instance Checking for $\mathcal{SROEL}(\times)$

We now present a calculus for instance checking – deciding if $C(a)$ is entailed for $C \in \mathbf{N_C}$, $a \in \mathbf{N_I}$ – for $\mathcal{SROEL}(\times)$. In

$$\begin{array}{rlrl}
C(a) \mapsto & \mathsf{SubClass}(a, C) & R(a, b) \mapsto & \mathsf{Ex}^-(a, R, b, b) \\
A \sqsubseteq C \mapsto & \mathsf{SubClass}(A, C) & A \sqcap B \sqsubseteq C \mapsto & \mathsf{Conj}^-(A, B, C) \\
\top \sqsubseteq C \mapsto & \mathsf{Top}(C) & A \sqsubseteq \exists R.\mathsf{Self} \mapsto & \mathsf{Self}^+(A, R) \\
A \sqsubseteq \bot \mapsto & \mathsf{Bot}(A) & \exists R.\mathsf{Self} \sqsubseteq C \mapsto & \mathsf{Self}^-(R, C) \\
\exists R.A \sqsubseteq C \mapsto & \mathsf{Ex}^-(R, A, C) & \{a\} \sqsubseteq C \mapsto & \mathsf{SubClass}(a, C) \\
A \sqsubseteq \exists R.B \mapsto & \mathsf{Ex}^+(A, R, B, \mathsf{e}^{A \sqsubseteq \exists R.B}) & A \sqsubseteq \{c\} \mapsto & \mathsf{SubClass}(A, c) \\
R \sqsubseteq T \mapsto & \mathsf{SubRole}(R, T) & R \sqsubseteq C \times D \mapsto & \mathsf{RProd}^+(R, C, D) \\
R \circ S \sqsubseteq T \mapsto & \mathsf{RChain}^-(R, S, T) & A \times B \sqsubseteq R \mapsto & \mathsf{RProd}^-(A, B, R) \\
\end{array}$$

$$a \in \mathbf{N_I} \mapsto \mathsf{Nom}(a) \qquad A \in \mathbf{N_C} \mapsto \mathsf{Cls}(A) \qquad R \in \mathbf{N_R} \mapsto \mathsf{Rol}(R)$$

$$A, B, C, D \in \mathbf{N_C}, R, S, T \in \mathbf{N_R}, a, b, c \in \mathbf{N_I}$$

Figure 1: Input translation I_{inst}

Section 5 we show its optimality in terms of certain characteristics. This study of calculi requires a uniform presentation for deduction calculi that have been proposed for $\mathcal{EL}$-type DLs, e.g., by Baader *et al.* [2005] and Delaitre and Kazakov [2009]. This motivates our use of Datalog in this section.

Intuitively speaking, a materialisation calculus is a system of deduction rules for deriving logical consequences. As opposed to a complete inference algorithm, it does not specify a control flow for evaluating these rules. Deduction rules can be denoted in many forms, e.g. using textual if-then descriptions [Baader *et al.*, 2005], in tabular form [Motik *et al.*, 2009], or as sequent calculus style derivation rules [Delaitre and Kazakov, 2009]. Premises and conclusions of rules often consist of logical formulae, but may also contain auxiliary expressions. Baader *et al.* [2005], e.g., use auxiliary statements $A \rightsquigarrow_R B$ for $A, B \in \mathbf{N_C}$. A deduction rule can then be viewed as a schema for deriving new expressions from a finite set of given expressions. In particular, the applicability of rules is normally not affected by uniform renamings of signature symbols in premise and conclusion.

Deduction rules in this sense can be denoted as Datalog rules where concrete logical sentences are represented as ground facts that use signature symbols in term positions. For example, we can represent $A \sqsubseteq B$ as $\mathtt{subclassOf}(A, B)$, and introduce a rule $\mathtt{subclassOf}(x, y) \wedge \mathtt{subclassOf}(y, z) \rightarrow \mathtt{subclassOf}(x, z)$. This unifies the presentation of calculi, and lets us exploit techniques from deductive databases. For connecting Datalog to DL, we require an input translation from individual DL axioms to (sets of) Datalog EDB facts. This translation is also defined for signature symbols, since symbols must generally be "loaded" into Datalog to be able to derive conclusions about them, regardless of whether the symbols occurred in input axioms or not. A formalisation of these ideas is given later in Definition 1.

Rule-based calculi suggest materialisation-based (or consequence-driven) reasoning: after translating a knowledge base to Datalog facts, all consequences of these facts under the deduction rules can be computed in a bottom-up fashion, and all supported entailments can then be checked without further recursive computation. This contrasts with other reasoning principles such as the tableaux method where typically just a single entailment is checked in one run of the algorithm.

It is not hard to formulate the deduction algorithms that were presented for $\mathcal{EL}$-type logics in [Baader *et al.*, 2005] and [Delaitre and Kazakov, 2009] using Datalog rules. The calculus we present here, however, is derived from a Data-

(1)	$\mathsf{Nom}(x) \to \mathsf{isa}(x,x)$
(2)	$\mathsf{Nom}(x) \wedge \mathsf{spo}(x,v,x) \to \mathsf{self}(x,v)$
(3)	$\mathsf{Top}(z) \wedge \mathsf{isa}(x,z') \to \mathsf{isa}(x,z)$
(4)	$\mathsf{Bot}(z) \wedge \mathsf{isa}(u,z) \wedge \mathsf{isa}(x,z') \wedge \mathsf{Cls}(y) \to \mathsf{isa}(x,y)$
(5)	$\mathsf{SubClass}(y,z) \wedge \mathsf{isa}(x,y) \to \mathsf{isa}(x,z)$
(6)	$\mathsf{Conj}^-(y_1,y_2,z) \wedge \mathsf{isa}(x,y_1) \wedge \mathsf{isa}(x,y_2) \to \mathsf{isa}(x,z)$
(7)	$\mathsf{Ex}^-(v,y,z) \wedge \mathsf{spo}(x,v,x') \wedge \mathsf{isa}(x',y) \to \mathsf{isa}(x,z)$
(8)	$\mathsf{Ex}^-(v,y,z) \wedge \mathsf{self}(x,v) \wedge \mathsf{isa}(x,y) \to \mathsf{isa}(x,z)$
(9)	$\mathsf{Ex}^+(y,v,z,x') \wedge \mathsf{isa}(x,y) \to \mathsf{spo}(x,v,x')$
(10)	$\mathsf{Ex}^+(y,v,z,x') \wedge \mathsf{isa}(x,y) \to \mathsf{isa}(x',z)$
(11)	$\mathsf{Self}^-(v,z) \wedge \mathsf{self}(x,v) \to \mathsf{isa}(x,z)$
(12)	$\mathsf{Self}^+(y,v) \wedge \mathsf{isa}(x,y) \to \mathsf{self}(x,v)$
(13)	$\mathsf{SubRole}(v,w) \wedge \mathsf{spo}(x,v,x') \to \mathsf{spo}(x,w,x')$
(14)	$\mathsf{SubRole}(v,w) \wedge \mathsf{self}(x,v) \to \mathsf{self}(x,w)$
(15)	$\mathsf{RChain}^-(u,v,w) \wedge \mathsf{spo}(x,u,y) \wedge \mathsf{spo}(y,v,z) \to \mathsf{spo}(x,w,z)$
(16)	$\mathsf{RChain}^-(u,v,w) \wedge \mathsf{self}(x,u) \wedge \mathsf{spo}(x,v,x') \to \mathsf{spo}(x,w,x')$
(17)	$\mathsf{RChain}^-(u,v,w) \wedge \mathsf{spo}(x,u,x') \wedge \mathsf{self}(x',v) \to \mathsf{spo}(x,w,x')$
(18)	$\mathsf{RChain}^-(u,v,w) \wedge \mathsf{self}(x,u) \wedge \mathsf{self}(x,v) \to \mathsf{spo}(x,w,x)$
(19)	$\mathsf{RProd}^-(y_1,y_2,w) \wedge \mathsf{isa}(x,y_1) \wedge \mathsf{isa}(x',y_2) \to \mathsf{spo}(x,w,x')$
(20)	$\mathsf{RProd}^-(y_1,y_2,w) \wedge \mathsf{isa}(x,y_1) \wedge \mathsf{isa}(x,y_2) \to \mathsf{self}(x,w)$
(21)	$\mathsf{RProd}^+(v,z_1,z_2) \wedge \mathsf{spo}(x,v,x') \to \mathsf{isa}(x,z_1)$
(22)	$\mathsf{RProd}^+(v,z_1,z_2) \wedge \mathsf{self}(x,v) \to \mathsf{isa}(x,z_1)$
(23)	$\mathsf{RProd}^+(v,z_1,z_2) \wedge \mathsf{spo}(x,v,x') \to \mathsf{isa}(x',z_2)$
(24)	$\mathsf{RProd}^+(v,z_1,z_2) \wedge \mathsf{self}(x,v) \to \mathsf{isa}(x,z_2)$
(25)	$\mathsf{isa}(x,y) \wedge \mathsf{Nom}(y) \wedge \mathsf{isa}(x,z) \to \mathsf{isa}(y,z)$
(26)	$\mathsf{isa}(x,y) \wedge \mathsf{Nom}(y) \wedge \mathsf{isa}(y,z) \to \mathsf{isa}(x,z)$
(27)	$\mathsf{isa}(x,y) \wedge \mathsf{Nom}(y) \wedge \mathsf{spo}(z,u,x) \to \mathsf{spo}(z,u,y)$

Figure 2: Deduction rules P_{inst}

log reduction introduced in [Krötzsch *et al.*, 2008] for a rule language based on $\mathcal{EL}^{++}$. This approach can be modified to cover $\mathcal{SROEL}(\times)$ and to use a fixed set of Datalog rules to yield a materialisation calculus in our sense. For simplicity, the following calculus only considers $\mathcal{SROEL}(\times)$ axioms of the basic forms in Fig. 1 (left of $\mapsto$). Other $\mathcal{SROEL}(\times)$ axioms can easily be normalised in linear time while preserving the original entailments [Krötzsch, 2010]. For a normalised knowledge base KB, we define a Datalog theory $P(\text{KB}) := P_{\text{inst}} \cup \{I_{\text{inst}}(\alpha) \mid \alpha \in \text{KB}\} \cup \{I_{\text{inst}}(s) \mid s \in \mathbf{N_I} \cup \mathbf{N_C} \cup \mathbf{N_R}\}$, where P_{inst} are the deduction rules in Fig. 2. This gives rise to a derivation calculus K_{inst}: we say that K_{inst} derives a fact $C(a)$ from KB if $P(\text{KB})$ entails $\mathsf{isa}(a, C)$.

Theorem 1 *The calculus K_{inst} is sound and complete.*

The IDB predicates isa, spo, and self in P_{inst} correspond to ABox axioms for atomic concepts, roles, and concepts $\exists R.\mathsf{Self}$, respectively. Rule (1) serves as an initialisation rule that accounts for the first isa facts to be derived. Rule (2) specifies the (only) case where reflexive spo facts lead to self facts. The rules (3) to (24) capture expected derivations for each of the axiom types as encoded by the EDB predicates. Rule (4) checks for global inconsistencies, and would typically not be materialised in implementations since its effect can directly be taken into account during entailment checking. Rules (9) and (10) make use of auxiliary constants $\mathsf{e}^{A \sqsubseteq \exists R.B}$ for handling existentials. Roughly speaking, each such constant represents the class of all role successors generated by the axiom from which it originates; see [Krötzsch, 2010] for details. The remaining rules (25) to (27) encode equality reasoning that is relevant in the presence of nominals where statements $\mathsf{isa}(a, b)$ with $a, b \in \mathbf{N_I}$ encode equality of a and b.

Axiom normalisation and the computation of I_{inst} can be accomplished in linear time, and the time for reasoning in Datalog is polynomial w.r.t. the size of the collection of ground facts. Using the known P-hardness of $\mathcal{EL}^{++}$ [Baader *et al.*, 2005], we obtain that instance checking in $\mathcal{SROEL}(\times)$ and in OWL EL without datatype properties is P-complete w.r.t. the size of the knowledge base. Strictly speaking, our treatment does not cover OWL EL *keys* [Motik *et al.*, 2009]. Conceived as a special form of *DL-safe rules*, they are easy to incorporate into our rule-based approach. OWL EL datatype properties could be treated like abstract roles in our algorithm, but with a reduced set of expressive features. This is possible since all EL datatypes are *convex* in the sense of [Baader *et al.*, 2005]. The main implementation burden is to evaluate the syntactic forms of datatype constants in the various OWL EL datatypes. The only additional feature is *functional datatype properties* for which another inference rule is required. The main work here is to observe that a naive implementation does not lead to wrong inferences, in contrast to functional concrete roles which cannot be captured by a rule-based calculus in our sense.

4 Classification in $\mathcal{SROEL}(\times)$

The materialisation calculus K_{inst} of Theorem 1 solves the instance checking problem for $\mathcal{SROEL}(\times)$. A calculus for checking satisfiability is easily derived since a $\mathcal{SROEL}(\times)$ knowledge base is inconsistent if and only if K_{inst} infers a fact $\mathsf{isa}(x, z)$ where $\mathsf{Bot}(z)$ holds. In this section, we ask how to obtain calculi for *classification* – the computation of all subsumptions of atomic classes implied by a knowledge base.

Class subsumption, too, can be reduced to instance retrieval: $\text{KB} \models A \sqsubseteq B$ holds if $\text{KB} \cup \{A(c)\} \models B(c)$ for a fresh c. This reduction requires the knowledge base to be modified, leading to new entailments, possibly even to global inconsistency. Thus K_{inst} cannot directly be used for classification. Rather, one needs a separate run of K_{inst} for each assumption $A(c)$ to compute all entailments of the form $A \sqsubseteq B$.

One can derive a materialisation calculus for classification in $\mathcal{SROEL}(\times)$ by "internalising" the runs of K_{inst}, extending all IDB predicates with an extra parameter to encode the test assumption under which an entailment holds. The name of c is not essential in assumptions $A(c)$, so one can simply re-use the Datalog constant A as the test instance of class A (Datalog does not care about the sort of A in DL). Following this discussion, it is straightforward to obtain a sound and complete classification calculus [Krötzsch, 2010].

This calculus is not very efficient since deductions that are globally true are inferred under each test assumption $A(c)$. So the number of globally derived facts can multiply by the number of class names, which can easily be 10^5 or more. This increase is reflected in our formalisation of materialisation calculi: the maximal arity of derived predicates would now be 4 while it had been 3 in K_{inst}, leading to potentially higher space requirements for materialised derivations. Implementations may achieve lower space bounds by suitable optimisations. Yet standard implementation techniques for Datalog, such as semi-naive materialisation, are sensitive to the arity of IDB predicates. In developing the *Orel* reasoner [Krötzsch *et al.*, 2010], we also experienced major *time* penalties for higher arities due to the larger numbers of inferences considered in each derivation step.

The maximal arity of IDB predicates thus is an important measure for the efficiency of a materialisation calculus. We call this the *arity of a calculus* and speak of binary/ternary/*n*-ary materialisation calculi. The search for efficient materialisation calculi can thus be formalised as the task of finding a ternary or binary calculus that is sound and complete for $\mathcal{SROEL}(\times)$ classification. Unfortunately, as shown in Section 5, no such calculus exists. To show that this is not obvious, we now give such a calculus for a slightly smaller DL.

We present a ternary classification calculus that supports role chains but no $\top$, $\bot$, nominals, and no concept products on the left-hand side of axioms. The input translation is as in Fig. 1 but restricted to the remaining features. So EDB predicates Top, Bot, and RProd^- are no longer used.

A set of rules is developed by restricting the rules to the remaining features. We refer to rules obtained from K_{inst} by the numbers in Fig. 2. Rules (3), (4), (19), and (20) are obsolete due to the omitted EDB predicates. Without nominals, we find that all derivations $\mathsf{isa}(x, y)$ are such that y is a DL class name, or y is a DL individual name and $x = y$. This is not hard to verify inductively by considering each rule, and the symbols used in relevant EDB facts. Therefore rules (25), (26), and (27) can be dropped. As shown in [Krötzsch, 2010], we do not need to introduce an extra parameter for keeping track of the assumption under which a subsumption was derived:

Theorem 2 *Consider the materialisation calculus K_{scc} with I_{scc} defined like I_{inst} in Fig. 1 but undefined for all axioms that use nominal classes, $\top$, $\bot$, or concept products on the left-hand side, and the program P_{scc} consisting of the rules* (1), (2), (5)–(18), *and* (21)–(24) *of Fig. 2 together with a new rule $\mathsf{Cls}(z) \to \mathsf{isa}(z, z)$.*

For a knowledge base KB *such that $I_{scc}(\alpha)$ is defined for all $\alpha \in$* KB, *set* $P(\text{KB}) := P_{scc} \cup \{I_{scc}(\alpha) \mid \alpha \in \text{KB}\} \cup \{I_{scc}(s) \mid s \in \mathbf{N_I} \cup \mathbf{N_C} \cup \mathbf{N_R}\}$*. Then for all $A, B \in \mathbf{N_C}$,* KB *entails $A \sqsubseteq B$ if and only if $P(\text{KB})$ entails $\mathsf{isa}(A, B)$, whenever $P(\text{KB})$ is defined. Thus K_{scc} provides a materialisation calculus for subsumption checking for $\mathcal{SROEL}(\times)$ knowledge bases that contain only $\sqcap$ (for concepts and roles), $\exists$, Self, $\circ$, and concept products on the right-hand side.*

This theorem covers all OWL EL ontologies without datatype properties and any of `owl:Thing`, `owl:Nothing`, `owl:topObjectProperty`, `owl:bottomObjectProperty`, `ObjectHasValue`, `ObjectOneOf`, and `HasKey`.

If no role chains occur, one can further simplify K_{scc} to obtain a binary classification calculus for normalised $\mathcal{SROEL}(\times)$ knowledge bases that contain only $\sqcap$ (for concepts and roles), $\exists$, Self, and concept products on the right-hand side. This is spelled out in [Krötzsch, 2010]. Delaitre and Kazakov [2009] used a similar approach to optimise a classification calculus for $\mathcal{ELH}$.

5 Minimal Arities of Materialisation Calculi

The arities of the above calculi for $\mathcal{SROEL}(\times)$ range from 2 to 4. We argued that low arities are important for efficiency, so one should develop calculi of minimal arity. Next, we establish lower bounds on the arity of materialisation calculi for various reasoning problems. We formalise materialisation calculi to generalise the calculi discussed above:

Definition 1 *A* materialisation calculus *K is a tuple $K = \langle I, P, O\rangle$ where I and O are partial functions, and P is a set of Datalog rules without constant symbols, such that*

1. *given an axiom or signature symbol α, $I(\alpha)$ is either undefined or a set of Datalog facts over EDB predicates,*
2. *given an axiom α, $O(\alpha)$ is either undefined or a Datalog fact over an IDB predicate,*
3. *the set of EDB and IDB predicates used by I, P, and O is fixed and finite,*
4. *all constant symbols used in $I(\alpha)$ or $O(\alpha)$ for some axiom (or signature symbol) α are either signature symbols that appear in (or are equal to) α, or constants of the form e_i^α with $i \geq 0$, where all constant names e_i^α are mutually distinct and unequal to any DL signature symbol,*
5. *I and O do not depend on concrete signature symbols, i.e. for a renaming ρ of signature symbols that maps individual/concept/role names to individual/concept/role names, we find $I(\rho(\alpha)) = \rho(I(\alpha))$ and $O(\rho(\alpha)) = \rho(O(\alpha))$ if $\rho(\mathsf{e}_i^\alpha) = \mathsf{e}_i^{\rho(\alpha)}$.*

For knowledge bases KB *we set* $I(\text{KB}) := \bigcup_{\beta\in\text{KB}} I(\beta)$ *if $I(\beta)$ is defined for all $\beta \in$* KB *and undefined otherwise. We extend I to sets of signature symbols S by setting* $I(S) := \bigcup_{s\in S, I(s)\text{ defined}} I(s)$. *$K$ induces an* entailment relation *$\vdash_K$ between knowledge bases* KB *and axioms α over a signature $\langle \mathbf{N_I}, \mathbf{N_C}, \mathbf{N_R}\rangle$, defined by setting* KB $\vdash_K \alpha$ *whenever* $I(\text{KB})$ *and $O(\alpha)$ are defined and* $I(\text{KB}) \cup I(\mathbf{N_I} \cup \mathbf{N_C} \cup \mathbf{N_R}) \cup P \models O(\alpha)$.

We say that K is sound (complete) *if* KB $\vdash_K \alpha$ *implies (is implied by)* KB $\models \alpha$ *for all knowledge bases* KB *and axioms α for which* $I(\text{KB})$ *and $O(\alpha)$ are defined.*

This allows the Datalog transformation I to introduce arbitrarily many auxiliary constants e_i^α. Syntactic normalisations that use auxiliary concept names could thus also be part of the translation. Yet, the input translation is limited, since it depends only on individual axioms and signature symbols. This precludes complex Datalog translations as in [Motik and Sattler, 2006; Rudolph *et al.*, 2008]. We make no assumptions on the computability or complexity of I and O, but both functions are typically very simple.

Now our general proof strategy is as follows. For a contradiction, we suppose a materialisation calculus of lower arity for a given reasoning problem. We then consider a particular instance of that problem, given by a knowledge base KB that entails some consequence α. Since the calculus is assumed to be complete, we find an according Datalog derivation with a corresponding proof tree. This proof tree is then modified by renaming constants, leading to a variant of the proof tree that is still valid for the given materialisation calculus, but based on different (renamed) assumptions. The modified assumptions correspond to a modified knowledge base KB′, where we find that the materialisation calculus still derives α on the input KB′. We then show that KB $\not\models \alpha$, so that the calculus

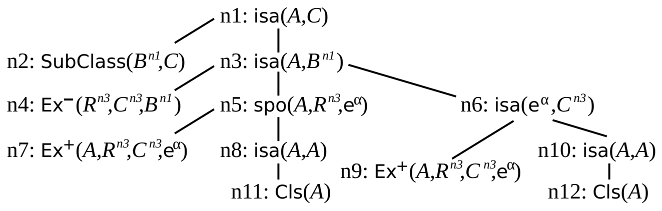
n1: isa(A,C)
n2: SubClass(B^{n1},C)
n3: isa(A,B^{n1})
n4: Ex$^-$(R^{n3},C^{n3},B^{n1})
n5: spo(A,R^{n3},e$^\alpha$)
n6: isa(e$^\alpha$,C^{n3})
n7: Ex$^+$(A,R^{n3},C^{n3},e$^\alpha$)
n8: isa(A,A)
n9: Ex$^+$(A,R^{n3},C^{n3},e$^\alpha$)
n10: isa(A,A)
n11: Cls(A)
n12: Cls(A)

Figure 3: A diversified K_{scc} proof (α denotes $A \sqsubseteq \exists R^{n3}.C^{n3}$)

cannot be sound. Some graph theoretic arguments are used to establish this last step. The essential modification of proof trees works as follows:

Definition 2 *Consider a materialisation calculus $K = \langle I, P, O\rangle$, a knowledge base* KB *such that I*(KB) *is defined, and a proof tree $T = \langle N, E, \lambda\rangle$ for I*(KB)$\cup I(\mathbf{N_I}\cup\mathbf{N_C}\cup\mathbf{N_R})\cup P$. *We say that a DL signature symbol σ* occurs *in a ground atom F if F contains σ as a constant, or if F contains some auxiliary constant e_i^α such that σ occurs in α. The* interface *of a node $n \in N$ is the set of signature symbols that occur in $\lambda(n)$.*

The (labels of the) tree T can be diversified *recursively:*

- *replace all signature symbols s that do not occur in the interface of the root node by a fresh symbol s' that has not yet been used in T or in this construction,*
- *recursively diversify the subtrees below each of the direct child nodes of the root.*

So T is diversified by replacing some (occurrences of) signature symbols with fresh symbols. We use s^n to denote the symbol by which s is replaced in node n. The renaming may affect auxiliary constants by renaming symbols in the axioms that are part of their name.

Intuitively, diversification removes all re-use of constants in a proof tree that is not essential for applying the rules of P. This is captured by each node's *interface*: constants not in the interface of a rule application can be renamed uniformly below the node without affecting applicability of the rule. So the arity of a calculus determines the amount of renaming during diversification. Figure 3 shows a diversification of a proof for $\{A \sqsubseteq \exists R.C, \exists R.C \sqsubseteq B, B \sqsubseteq C\} \models A \sqsubseteq C$ in the calculus K_{scc} of Theorem 2. Note how C is renamed to C^{n3} in some labels only. No further renamings occur below the nodes $n5$ and $n6$ since all relevant symbols occur in their interface due to the auxiliary constant e^α.

Leaf nodes n in proof trees relate to input symbols or axioms. In the latter case we find an axiom α_n such that $\lambda(n) \in I(\alpha_n)$. By suitably renaming symbols in the axioms α_n, one can find a *diversified knowledge base* for which the diversified proof tree encodes a valid derivation. The diversified knowledge base for Fig. 3, e.g., is $\{A \sqsubseteq \exists R^{n3}.C^{n3}, \exists R^{n3}.C^{n3} \sqsubseteq B^{n1}, B^{n1} \sqsubseteq C\}$, which clearly entails $A \sqsubseteq C$ as before.

The structure of a diversified proof tree T is mirrored in the corresponding diversified knowledge base KB. An axiom of form $\alpha_n \in$ KB is *below* a node m if n is below m, and we set $\text{KB}_m := \{\alpha_n \in \text{KB} \mid n \text{ below } m\}$. Diversification ensures that symbols occurring in both KB_m and $\text{KB} \setminus \text{KB}_m$ must belong to the interface of m. This interface includes all DL symbols in $\lambda(m)$. If auxiliary constants e^α occur, this encompasses *all* symbols of a given input axiom α. Yet, the arity limits the number of such axioms: for a calculus of arity a, the interface of any node can comprise no more than the set of DL symbols that occur in a axioms of the input knowledge base.

This can be interpreted graphically based on the *dependency graph* of KB – the graph that has the signature symbols in KB as its nodes, and, for each axiom of KB with exactly n symbols, an n-ary hyperedge connecting these n symbols. The set KB_m induces a subgraph of a dependency graph, and the interface of m describes the nodes that this subgraph is allowed to share with the remaining graph.

One can use this machinery to prove the following theorems. The key in each case is to show that a conclusion that a supposed materialisation calculus of lower arity produces is not entailed by a diversified knowledge base. Namely, for the entailment to hold, the latter would need to contain a set of axioms that cannot possibly be distributed over the structure of a proof tree without violating the interface constraints. See [Krötzsch, 2010] for detailed proofs.

Theorem 3 *Let $\mathcal{L}$ be a DL with general concept inclusions, existential quantification, and role chains. Every materialisation calculus that is sound and complete for classification or instance retrieval in $\mathcal{L}$ has arity three or more.*

Theorem 4 *Let $\mathcal{L}$ be a DL with general concept inclusions, existential quantification, and nominal classes. Every materialisation calculus that is sound and complete for classification in $\mathcal{L}$ has arity three or more.*

Theorem 5 *Let $\mathcal{L}$ be a DL with general concept inclusions, existential quantification, role chains, and nominal classes. Every materialisation calculus that is sound and complete for classification in $\mathcal{L}$ has arity four or more.*

The latter two results do not extend to instance retrieval, so in a sense classification is harder to implement efficiently. Indeed, Theorem 1 shows that a ternary instance retrieval calculus exists for a DL that includes existentials, nominals, and role chains. For DLs as in Theorem 4, we have not presented calculi of optimal arity. A ternary (binary) calculus for classification (instance retrieval) in this case can be obtained by using similar techniques as for the binary calculus $K_{\text{sc-}}$ presented in [Krötzsch, 2010]. Theorem 5 may be surprising, given that the $\mathcal{EL}^{++}$ calculus in [Baader *et al.*, 2005] would be ternary in our notation. The explanation is that this algorithm is incomplete for classification; the proof of Theorem 5 can be used to find a counter example [Krötzsch, 2010].

6 Datalog Width and Complexity Theory

The presented work focusses on practical tasks of reasoning in OWL EL, and studies the arity of materialisation calculi mainly as a tool for optimising inferencing systems. Yet, the work can be related to foundational studies in complexity theory and deductive databases.

It is known that the arity of IDB predicates affects Datalog expressivity in the sense that, for any natural number n, one can find problems that cannot be expressed in Datalog of IDB arity $\leq n$ [Afrati and Cosmadakis, 1989]. Arity in this work is called *width*, and has later been related to *persistency numbers* [Afrati *et al.*, 2005].

Feder and Vardi [1999] study Datalog in the context of complexity theory to classify the difficulty of certain constraint satisfaction problems. There a problem is said to have *width* (l, k) if it can be solved using a Datalog program of at most k variables per rule and at most l variables per head. In this sense, our work details how DL inferencing can be reduced to bounded width constraint satisfaction problems. The results of Section 5 then establish a minimal width of the constraint satisfaction problems that are obtained when restricting the initial translation as in Definition 1. This also connects to *parameterised complexity* since proof trees of (l, k) programs provide a tree decomposition of treewidth $k - 1$ for the dependency graph of EDB facts. Our arity, however, corresponds to l – the maximal number of elements that are shared between adjacent bubbles of a tree decomposition. An upper bound on k suffices to establish tractability, but the consideration of l as a practically relevant measure appears to be new.

7 Summary and Conclusions

We presented various inferencing calculi for $\mathcal{SROEL}(\times)$ and its fragments, including the first sound and complete polynomial time calculus for inferencing in a DL that largely captures the OWL EL ontology language. Using a simple framework for expressing materialisation calculi in Datalog, the arity of IDB predicates is an interesting measure for the worst-case space requirements of materialisation-based algorithms. We have shown that role chains and nominals cause this measure to increase, while other features do not have this impact.

Hence we can differentiate $\mathcal{SROEL}(\times)$ fragments and inferencing tasks based on a measure that relates to the efficiency of actual implementations. The findings agree with practical experiences that especially nominals and role chains are harder to implement efficiently than basic $\mathcal{EL}$ features. Results on worst-case complexity so far have not been able to explain such discrepancies, since all reasoning problems we consider are P-complete. Conversely, some advanced features of OWL EL do not seem to make inferencing any harder. Overall, such results can guide the decision which features to implement or to use in an application.

Although there are standard implementation strategies for Datalog, these insights are independent of actual algorithms. The design and implementation of optimised "pay as you go" strategies for EL inferencing is an important goal that we will further pursue in developing the ELK reasoner[1]. Furthermore, we conjecture that our results about Datalog arity can be strengthened to obtain more direct statements about space complexity of almost arbitrary monotone calculi.

Acknowledgements The author thanks Yevgeny Kazakov and the anonymous reviewers for valuable input. This work was supported by DFG in project *ExpresST* and by EPSRC in project *ConDOR* (EP/G02085X/1).

References

[Afrati and Cosmadakis, 1989] Foto N. Afrati and Stavros S. Cosmadakis. Expressiveness of restricted recursive queries. In *STOC'89*, pages 113–126. ACM, 1989.

[Afrati *et al.*, 2005] Foto N. Afrati, Stavros S. Cosmadakis, and Eugénie Foustoucos. Datalog programs and their persistency numbers. *ACM Trans. Comput. Log.*, 6(3):481–518, 2005.

[Baader *et al.*, 2005] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. In *IJCAI'05*, pages 364–369. Professional Book Center, 2005.

[Baader *et al.*, 2006] Franz Baader, Carsten Lutz, and Boontawee Suntisrivaraporn. CEL—a polynomial-time reasoner for life science ontologies. In *IJCAR'06*, volume 4130 of *LNCS*, pages 287–291. Springer, 2006.

[Baader *et al.*, 2007] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, second edition, 2007.

[Baader *et al.*, 2008] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope further. In *OWLED 2008 Workshop*, volume 496 of *CEUR*, 2008.

[Delaitre and Kazakov, 2009] Vincent Delaitre and Yevgeny Kazakov. Classifying $\mathcal{ELH}$ ontologies in SQL databases. In *OWLED 2009 Workshop*, volume 529 of *CEUR*, 2009.

[Feder and Vardi, 1999] Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory. *SIAM J. Comput.*, 28:57–104, 1999.

[Hitzler *et al.*, 2009] Pascal Hitzler, Markus Krötzsch, and Sebastian Rudolph. *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC, 2009.

[James and Spackman, 2008] Andrew G. James and Kent A. Spackman. Representation of disorders of the newborn infant by SNOMED CT. In *Proc. 21st Int. Congress of the European Federation for Medical Informatics (MIE'08)*, pages 833–838, 2008.

[Krötzsch *et al.*, 2008] Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. ELP: Tractable rules for OWL 2. In *ISWC'08*, volume 5318 of *LNCS*, pages 649–664. Springer, 2008.

[Krötzsch *et al.*, 2010] Markus Krötzsch, Anees Mehdi, and Sebastian Rudolph. Orel: Database-driven reasoning for OWL 2 profiles. In *23rd Int. Workshop on Description Logics (DL'10)*, 2010.

[Krötzsch, 2010] Markus Krötzsch. Efficient inferencing for OWL EL. In *JELIA'10*, volume 6341 of *LNAI*, pages 234–246. Springer, 2010. Extended technical report available at `http://www.aifb.kit.edu/web/Techreport3005`.

[Motik and Sattler, 2006] Boris Motik and Ulrike Sattler. A comparison of reasoning techniques for querying large description logic ABoxes. In *LPAR'01*, volume 4246 of *LNCS*, pages 227–241. Springer, 2006.

[Motik *et al.*, 2009] Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz, editors. *OWL 2 Web Ontology Language: Profiles*. W3C Recommendation, 2009. Available at `http://www.w3.org/TR/owl2-profiles/`.

[OWL, 2009] W3C Working Group OWL. *OWL 2 Web Ontology Language: Document Overview*. W3C Recommendation, 2009. Available at `http://www.w3.org/TR/owl2-overview/`.

[Rudolph *et al.*, 2008] Sebastian Rudolph, Markus Krötzsch, and Pascal Hitzler. Description logic reasoning with decision diagrams: Compiling $\mathcal{SHIQ}$ to disjunctive datalog. In *ISWC'08*, volume 5318 of *LNCS*, pages 435–450. Springer, 2008.

[1]`http://code.google.com/p/elk-reasoner/`